const & with classes

Ch 11.3
Highlights

- `const &` passing-by-reference

```c
void foo(person a, const person& b);
```
An object is simply a box in memory and if you pass this into a function it makes a copy.

A memory address is where a box is located and if you pass this into a function, you can change the variable everywhere.

<table>
<thead>
<tr>
<th>Memory address</th>
<th>Object (box)</th>
</tr>
</thead>
<tbody>
<tr>
<td>arrays</td>
<td>int, double, char, ...</td>
</tr>
<tr>
<td>using &amp;</td>
<td>classes</td>
</tr>
</tbody>
</table>
What is the difference between these two?

\begin{verbatim}
int sum(int x, int y);
int sum(const int &x, const int &y);
\end{verbatim}
const call-by-reference

What is the difference between these two?

```cpp
int sum(int x, int y);
int sum(const int &x, const int &y);
```

First one copies the values into x and y, thus these values exist in multiple places.

The second creates a link but does not let you modify the original (see: callByValue.cpp)
Classes can be rather big, so in this case using `const` and `&` can save memory.

So a better way to write:

```cpp
bool equals(Point first, Point second)
```

... would be: (function definition the same)

```cpp
bool equals(const Point & first, const Point & second)
```

In fact, without `&` creates a copy, which is a new object and thus runs a constructor.
Average class

Let's do a larger example of a class (like a lab)

Suppose we want to make a class to find the average of an array of doubles, and want:
- A default constructor
- A constructor that takes an array and size
- Overload ‘<<’ for cout and average
- Overload ‘+=’ to average over another double
- Overload ‘+’ to add two averages together
(See: average.cpp)
typedef

Side note: If you want to rename types, you can do that with `typedef` command:

```cpp
typedef int DefinatelyNotAnInt;  
DefinatelyNotAnInt x;  
x=3;  
int y = x;  
cout << y;
```

If you have always been bothered that we use “double” instead of “real”, go ahead and fix it!
Simple card game

Let's make a small card game based on hearthstone (simplified)
Simple card game

Rules:
- Cards **attack** and **defense** values
- Defender draws one card
- All cards always attack
- Defender selects which card to defend with
- Attacker dies if attack < defense (versa versa)
(See: simpleStone.cpp)